

Как заставить Codex| работать правильно

Постановка задач,
контроль и проверка
результата

ВИКТОР КРОСС

Виктор Кросс

Как заставить Codex работать правильно.

Постановка задач, контроль и проверка результата

*<https://litres.ru/74585173>
SelfPub; 2026*

Аннотация

Codex может написать правильный код и все равно провалить вашу задачу. Самая опасная ошибка ИИ выглядит не как ошибка: тесты зеленые, отчет красивый, система пишет «готово», а пользователь не получил нужный результат. Эта книга учит не сочинять «магические промпты», а управлять Codex как исполнителем. Вы получите договор о задаче из семи элементов, правила работы с фактами и источниками истины, матрицу полномочий, минимальные срезы, контрольные точки и многоуровневую приемку. Реальные кейсы показывают, как ложное завершение рождается в коде, приложениях, SEO, MCP и действиях на сайтах. Плюс 12 готовых шаблонов. Если Codex уже экономит вам часы, эта книга поможет не потерять дни на переделке того, что он уверенно назвал законченным. При редакционной подготовке текста использовался Gemini.

Содержание

Введение	9
Что дают шесть проектов	12
Ключевое суждение книги	14
Человек по-прежнему незаменим	15
Руководство по чтению: примените эту книгу в следующем задании	16
Если Codex часто отклоняется от темы	17
Если задания всегда приходится переделывать на полпути	18
Если система всегда сообщает о завершении, но вы не уверены	19
Если Codex будет работать с реальным сайтом или учетной записью	20
Если вы собираетесь создавать систему для долгосрочного сотрудничества	21
В каждой задаче выполняйте только три действия	22
Как использовать приложения	23
Рекомендуемый подход	25
Часть 1. Переосмысление отношений между человеком и Codex	26
Глава 1. Codex не боится сложности, он боится неясных целей	27

Когда «создать ещё 64 страницы» кажется вполне разумным	28
Неясность не означает краткость	30
Codex ищет реализуемые альтернативные цели	32
Четыре наиболее распространённых вида неясности целей	35
Неизвестно, какое состояние нужно изменить	36
Неизвестно, что важнее чего	37
Непонятно, когда остановиться	38
Сначала опишите результат, а потом — функциональность	39
Пусть Codex сначала продемонстрирует своё понимание	41
Ясно ли сформулирована цель: самопроверка с помощью шести вопросов	43
Основания и ограничения данной главы	45
Глава 2. Он написал правильный код, почему же он всё равно поступил неправильно	46
Неправильный ответ, получивший «12 баллов из 12»	47
«Сделать правильно» на самом деле имеет три уровня	49
Почему Codex так серьёзно ошибается	51
1. Люди задавали функциональные цели, но не задавали цели, связанные с игровым	52

опытом	
2. Локальные документы стали высшим авторитетом в вопросах ошибок	54
3. Критерии приемки измеряют то, что можно измерить, но не измеряют то, что важно	55
4. Люди слишком поздно видят реальные результаты	56
По-настоящему эффективное исправление отклонений — это не просто сказать: «Сделайте проще»	57
От простого требования к договору о выполнении задания	62
1. Целевой результат	63
2. Текущие факты и неизвестные	64
3. Авторитетный контекст	65
4. Объем и нецелевые задачи	66
5. Полномочия и риски	67
6. Этапы выполнения и контрольные точки	68
7. Доказательства приемки и условия остановки	69
Когда исправлять, а когда начинать с нуля	70
Что должно подтверждать автоматическое тестирование, а что — нет	72
Эта проблема характерна не только для игровых проектов	74
Правильное распределение обязанностей	76

между человеком и Codex	
Готово к использованию: трёхуровневая проверка правильности	79
Правильность ценности	80
Правильность задачи	81
Правильная реализация	82
Совет по началу работы, который можно скопировать напрямую	83
Основания и ограничения данной главы	85
Глава 3. Codex — исполнитель, человек — руководитель	87
Вход в админ-панель означает, что можно приступать к работе?	88
Управляющий — не самый занятой человек	90
Разделение полномочий на три категории	92
Право на внесение предложений	93
Право на исполнение	94
Право принятия решений	95
Карта полномочий, более полезная, чем «полная автоматизация»	96
Не следует дистанционно контролировать шаги, а нужно управлять соглашением	98
Четыре категории решений, которые нельзя передать на аутсорсинг	100
Принятие рисков	101
Достаточность доказательств	102

Компромиссы	103
Какие обязанности Codex должен брать на себя по собственной инициативе	104
Можно использовать напрямую: карта полномочий ответственного лица	106
Самопроверка ответственного лица	108
Основания и границы данной главы	110
Глава 4. Подсказки — это лишь отправная точка, а не полноценный метод	111
Почему даже очень длинное задание всё равно может сойти с курса	112
На что способен промпт	114
Почему одноразовый чат не работает	116
План не обновляется с учётом фактических данных	117
В объявлении о завершении отсутствует возможность опровержения	118
Отсутствие накопления опыта	119
Переход от подсказок к пятиуровневой системе задач	120
Второй уровень: договор о задаче и план выполнения	121
Третий уровень: постоянные правила проекта	122
Четвёртый уровень: исполняемые ограничения	123
Пятый уровень: независимая проверка и	124

ретроспектива	
Где следует разместить тот или иной опыт	125
Различие между «общими подсказками» и «системой задач»	127
Как запустить долгосрочную задачу	129
Три вопроса, которые нужно задать после каждого исправления отклонения	131
Самопроверка системы задач	132
Основания и границы данной главы	134
Часть 2: Превращение идей в выполнимые задачи	135
Глава 5. Не начинайте со списка функций	136
Список, выглядящий очень профессионально	137
Почему список функций так привлекателен	139
Формулируйте требования в виде изменений состояния	140
Целевое состояние должно также включать состояние неудачи	142
Сначала спросите: «Можно ли повторно использовать?», а потом: «Что нужно добавить?»	144
Конец ознакомительного фрагмента.	145

Виктор Кросс Как заставить Codex работать правильно. Постановка задач, контроль и проверка результата

Введение

Когда я впервые собирал эти примеры, мне тоже было легко понять проблему как «какие «ловушки» встречались Codex при написании кода».

В игре не сбрасывается состояние, настольное ПО проходит компиляцию, но не запускается, «Smoke Test» плаги-на запускается в неправильном каталоге, веб-агент случайно нажимает кнопку, а сайт с инструментами генерирует слишком много страниц, не имеющих самостоятельной ценности. Если перечислить все эти проблемы по пунктам, получится практическое руководство по устранению неисправностей.

Но между ними есть одна, более важная общая черта.

Зачастую Codex не то чтобы не выполняет задачу. Он пишет код, проводит тестирование, генерирует страницы, нажимает кнопки и даже может предоставить четко структурированный отчет о выполнении. Настоящее отклонение происходит на более раннем этапе: человек не определил, что в конечном итоге должен получить пользователь, не отделил текущие факты от старого плана, не указал, чье слово будет решающим в случае конфликта данных, не прописал, чего делать нельзя, и не обеспечил доступа инструментам приемочной проверки к реальному объекту поставки.

В результате неверные цели могут привести к созданию большого количества правильного кода, локальные «зеленые огни» могут быть интерпретированы как общее завершение, а статус входа в систему — как бизнес-авторизация.

Поэтому эта книга не сосредоточена на «универсальных подсказках».

Конечно, подсказки важны. Четкое описание цели, контекста, ограничений и условий завершения может значительно уменьшить количество недоразумений. Но по-настоящему надёжное использование Codex также включает: договор о задаче, реальную базовую линию, авторитетные данные, нецели, матрицу полномочий, минимальный разрез, реальное выполнение, поэтапное завершение, ручное утверждение и закрепление результатов в ходе ретроспективы.

Другими словами, мы обсуждаем не то, как сказать ИИ что-то более умное, а то, как построить сотрудничество, спо-

собное работать в долгосрочной перспективе.

Что дают шесть проектов

Книга «Игра разума» демонстрирует, почему даже уровень, получивший максимальный внутренний балл, всё равно приходится переделывать: тестирование подтверждает старые цели, но человек должен решить, стоит ли играть в такую игру.

PixCloak демонстрирует, как реальный поиск данных изменил задачу с «сделать ещё 64 страницы» на утилизацию старых запасов, что в итоге привело к сокращению на 168 индексных страниц.

ClipDock демонстрирует, как «более полная функциональность» превратилась во второй поисковый продукт, а также показывает разрыв в доказательствах, когда созданный «полностью зелёный» реальный EXE-файл не открывается.

Market Intelligence Core и Platform продемонстрировали, почему исходный код Smoke, кэш установки, нативный MCP, реальный Canary, Production и потребности пользователей представляют собой разные проблемы, а также показали, как независимая проверка опровергает полный «зелёный» статус общего доступа.

Интернет-магазин Jingmai продемонстрировал проблемы, возникающие после нажатия кнопки агентом: равнозначны ли вход в систему и авторизация, равнозначно ли появле-

ние всплывающего окна об успешном выполнении операции завершению бизнес-процесса, можно ли повторить попытку при неопределенном состоянии веб-страницы, и на ком в конечном итоге лежит ответственность за ведение бизнеса.

Эти примеры — не шесть историй успеха. В них сохранены неудачи, доработки, не подтвержденные результаты и внешние ограничения доступа. Именно поэтому они демонстрируют подход к реальному сотрудничеству, а не пропагандируют иллюзию автоматизации без каких-либо трений.

Ключевое суждение книги

Хорошее проектирование задач не гарантирует, что Codex никогда не будет ошибаться; оно позволяет обнаруживать ошибки раньше, когда они ещё незначительны, легче их выявлять и исправлять, а также предотвращает ситуацию, когда частичный успех выдается за полное завершение.

На протяжении всей книги этот вывод воплощается в виде «Договора о задаче», состоящего из семи элементов:

- целевой результат;

- текущие факты и неизвестные;

- контекст полномочий;

- Объем и нецелевые задачи;

- полномочия и риски;

- Этапы выполнения и контрольные точки;

- доказательства приемки и условия прекращения.

Это не бюрократические формы, которые необходимо заполнять пункт за пунктом. Задачи с низким уровнем риска можно свести к нескольким предложениям, тогда как задачи с высоким уровнем риска и длительным циклом требуют полного развертывания. Критерием оценки всегда является следующий вопрос: отсутствие какой информации привело бы к иному результату в Codex?

Человек по-прежнему незаменим

Codex может считывать большие объёмы документации, отслеживать межмодульные взаимосвязи, проводить проверки, формировать черновые версии и сохранять доказательства. Он может взять на себя рутинную работу, для которой раньше требовалась координация действий нескольких специалистов.

Однако именно человек должен решать, какие эксперименты стоит проводить, какие риски приемлемы, какие внешние действия разрешены, какие доказательства достаточны для выпуска, а также продолжает ли проект решать реальные проблемы.

Ценность человека заключается не в том, чтобы построчно указывать Codex, как писать код, а в том, чтобы принимать решения на нужном уровне.

Если эта книга поможет читателю реже говорить «продолжаем оптимизацию» и чаще задавать вопросы «что в конечном итоге должен сделать пользователь, каковы текущие факты, какие доказательства могут это подтвердить», то она выполнила свою задачу.

Руководство по чтению: примените эту книгу в следующем задании

Эту книгу можно читать от начала до конца, а можно выбрать путь в зависимости от проблем, с которыми вы сталкиваетесь в данный момент.

Если Codex часто отклоняется от темы

сначала прочтите главы 1, 5, 7, 8 и 10.

Вы поочерёдно рассмотрите целевые результаты, список функций, авторитетные данные, нецелевые задачи и полное задание. Не спешите искать более длинные подсказки — сначала проверьте, не оптимизирует ли Codex какой-то промежуточный показатель, который вам на самом деле не нужен.

Если задания всегда приходится переделывать на полпути

Сначала прочтите главы 2, 11, 12, 14 и 15.

Главное: разбейте большую цель на выполнимые части, сначала соберите факты, а потом планируйте, исправляйте отклонения с помощью реальной обратной связи и различайте локальные недостатки и ошибки в направлении.

Если система всегда сообщает о завершении, но вы не уверены

Сначала прочтите главы 13, 16, 17, 18, 19 и 20.

Вы создадите иерархию доказательств, разграничите статусы PASS, FAIL, PENDING, BLOCKED и UNKNOWN, а также разделите исходный код, установку, выполнение, платформу и бизнес-состояние.

Если Codex будет работать с реальным сайтом или учетной записью

сначала прочтите главы 3, 9, 16, 20, 21 и 26.

Перед началом работы создайте матрицу прав доступа, точные объекты, сравнения «до/после», бизнес-последствия и пакеты для ручного утверждения. Статус входа в систему не означает авторизацию, всплывающее окно об успешном выполнении не является конечным состоянием; не следует слепо повторять попытки, если побочные эффекты неопределенны.

Если вы собираетесь создавать систему для долгосрочного сотрудничества

сначала прочтите главы 4, 7, 10, 22 и приложение.

Размещайте разовые требования в подсказках и соглашениях о задачах, долгосрочные правила репозитория — в файле AGENTS.md, правила, поддающиеся автоматическому выполнению, преобразуйте в тесты или средства контроля доступа, а повторяющиеся задачи — в навыки (Skill) или стабильные рабочие процессы.

В каждой задаче выполняйте только три действия

Если у вас пока нет времени применять все методы, по крайней мере придерживайтесь следующих трёх шагов:

Перед началом работы попросите Codex повторить :
каковы, по его мнению, цель, объем и условия завершения;
перед пакетным выполнением просмотрите реальный срез: не копируйте направления, которые ещё не подтверждены;

В заключение необходимо предоставить многоуровневый отчет: что было сделано, что было доказано и что не было доказано.

Как использовать приложения

Приложение не предназначено для того, чтобы копировать его целиком за один раз. В зависимости от рисков задачи выберите минимальный шаблон:



Локальное исправление: контракт на легкие задачи + многоуровневый отчет о выполнении;



Новая функциональность: карта результатов пользователя + минимальная карта по вертикали;



Долгосрочный проект: полный контракт на задачу + карта авторитетных данных + промежуточная проверка;



Внешние операции: матрица полномочий + карта утверждения действий + тройная проверка;



Устранение инцидентов: UNKNOWN — сверка + карта анализа взаимодействия человека и машины;



Выпуск и сдача: список сдачи + форма утверждения «человек».

Скобки, квадратные скобки и пустые поля в шаблоне необходимо заменить. Не воспринимайте сам шаблон как

волшебную формулу; по-настоящему важны факты и суждения, которые вы в него вносите.

Рекомендуемый подход

Выберите реальную текущую задачу, а не вымышленное упражнение. Сначала составьте первую версию, используя договор о задаче из приложения, а затем позвольте Codex указать неизвестные факторы, которые могут изменить план. Завершив минимальный срез, вернитесь к главе 17 для проведения многоуровневого приемочного тестирования.

Если возникнут проблемы, не спешите сразу добавлять более жесткие указания. Сначала определите, к какому из следующих элементов относится пробел: цели, факты, контекст, объем, полномочия, срез или приемка, а затем внесите исправления в соответствующий уровень.

Ценность этой книги не в том, чтобы просто прочитать её, а в том, сможете ли вы и Codex в следующем задании раньше обнаружить то же самое недоразумение.

Часть 1. Переосмысление отношений между человеком и Codex

Пока не будем обсуждать приёмы и инструменты, давайте сначала ответим на более фундаментальный вопрос: за что должны отвечать человек и Codex, и почему технически правильная работа всё же может привести к неправильному результату.

Глава 1. Codex не боится сложности, он боится неясных целей

Суть этой главы: сложные задачи можно разбить на части, но неясные цели заставляют Codex продолжать оптимизацию по неверным показателям. Перед началом работы самое важное — не перечислять функции, а определить, чье состояние и как должно измениться.

Когда «создать ещё 64 страницы» кажется вполне разумным

Сайт, предлагающий инструменты для работы с изображениями и PDF-файлами, хочет привлечь больше трафика из поисковых систем. Проект уже обладает возможностями пакетного создания страниц, метаданных и многоязычного контента, а в стратегическом документе перечислены десятки новых возможностей для создания страниц с длинным хвостом.

Для Codex это очень удобная задача: входные данные чёткие, результат поддаётся подсчёту, алгоритмы можно повторно использовать, а по завершении можно отчитаться о том, сколько страниц, ключевых слов и языковых версий было добавлено.

Если бы задача звучала как «продолжить расширение SEO-страниц», Codex вполне способен был бы выполнить её на высоком уровне.

Проблема в том, что реальные данные поисковой системы указывают на другое направление: из 224 известных URL-адресов 140 не были проиндексированы. Возможно, сайту в тот момент не хватало не большего количества страниц, а меньшего количества более самостоятельных и заслуживающих внимания страниц.

«Количество новых страниц» измерить легко, а вот для

оценки «роста эффективного поискового инвентаря» приходится ждать внешних данных. В результате первое незаметно заменяет второе, становясь прокси-показателем проекта.

В этот момент чем эффективнее работал Codex, тем больше было растрачиваемых ресурсов.

Впоследствии задача изменилась с «продолжать генерировать» на «сначала прочитать реальное состояние индекса, определить, какие страницы следует сохранить, объединить, перенаправить, запретить к индексации или удалить». Из 202 URL-адресов, находившихся в старом онлайн-режиме, в итоге осталось 34 индексированных URL-адреса, а затем их число увеличилось до 37 только благодаря появлению трёх реальных рабочих процессов, имеющих самостоятельную ценность.

Этот поворот стал не просто корректировкой SEO-стратегии, а исправлением самого определения задачи: от «производить больше контента» к «улучшать реальные результаты, которые видят пользователи и поисковые системы».

Неясность не означает краткость

Многие считают, что краткая инструкция — это неясность, а длинная — ясность. На самом деле это не так.

Следующее предложение очень короткое, но при этом достаточно чёткое:

Исправить проблему, при которой приложение иногда не запускается при запуске из панели задач. Ориентироваться на запуск из реальной установленной версии; задача считается выполненной, если удалось 10 раз подряд запустить и закрыть приложение без остаточных процессов.

В нём указаны проблема, объект, сценарий и условия завершения.

Следующее предложение довольно длинное, но при этом остаётся неясным:

Пожалуйста, проведите комплексную оптимизацию сайта в отношении SEO, охвата контента, пользовательского опыта и монетизации. Опираясь на передовой опыт, добавляйте больше высококачественных страниц, совершенствуйте внутреннюю перелинковку и метаданные, сделайте сайт более профессиональным и конкурентоспособным, а также обеспечьте прохождение всех проверок.

В ней много прилагательных, но нет ответов на следующие вопросы: в чём заключается текущая проблема, каким категориям пользователей следует уделить приоритет-

ное внимание, какие данные подтверждают целесообразность создания дополнительных страниц, что нельзя менять и как проверяется так называемая «повышенная конкурентоспособность».

Таким образом, ясность задачи зависит не от количества слов, а от того, уменьшает ли она ключевую неопределённость, которая может изменить план действий.

Codex ищет реализуемые альтернативные цели

Когда настоящая цель человека не может быть реализована напрямую, Codex должен найти какое-то операциональное выражение.

«Сделать сайт более успешным» нельзя напрямую записать в код, поэтому это может быть представлено в виде большего количества страниц, более полных метаданных или более высокого охвата тестированием.

«Сделать программное обеспечение более похожим на продукт» нельзя реализовать напрямую, поэтому это может быть представлено в виде большего количества страниц, большего количества настроек, большего количества визуальных эффектов.

«Запустить работу интернет-магазина» нельзя оценить напрямую, поэтому это может быть представлено в виде успешных кликов, увеличения количества черновиков или завершенных отправленных форм.

Сами по себе эти замещающие показатели не являются ошибочными. Ошибка заключается в том, что люди не поясняют: они являются лишь одним из доказательств на пути к цели, а не самой целью.

В шести примерах неоднократно встречались подобные заменители:

Результат, который действительно волнует людей

Простые для оптимизации прокси-показатели

Игроки быстро понимают и получают удовольствие от процесса рассуждений

Панели, шаги, узлы доказательств и внутренние оценки

Инструментарий для получения эффективного поискового трафика

Количество страниц, ключевых слов и слов в контенте

Программное обеспечение действительно работоспособно и готово к поставке

Количество успешных компиляций, прохождений единичных тестов и отправленных версий

Плагины могут быть реально установлены и использованы пользователями

«Smoke Test» в репозитории разработки

Поведение платформы соответствует условиям бизнес-соглашения

Все показатели доступа — «зеленые»

Правильные товары находятся в правильном бизнес-ста-

туса

Успешное нажатие или появление уведомления об успешном выполнении

Проблема Codex обычно заключается не в отказе от выполнения задачи, а в том, что он продолжает работать при видимых условиях. Пока показатели прокси не привязаны к реальным результатам, он может стать очень усердным локальным оптимизатором.

Четыре наиболее распространённых вида неясности целей Неизвестно, для кого вносятся изменения

«Создать страницу поиска» и «помочь пользователю, у которого уже сохранены тысячи скриншотов, быстрее находить старый контент» приведут к совершенно разным решениям. В первом случае легко добавить второй набор страниц и команд, во втором — возможно, достаточно просто повторно использовать существующий основной холст и добавить фильтры.

Целевой пользователь — это не просто маркетинговый портрет, а реальные пользователи и сценарии, определяющие решение.

Неизвестно, какое состояние нужно изменить

«Сделать плагин» не описывает ни отправную точку, ни конечный результат. Речь идет о том, запустится ли исходный код или удастся ли установить установочный пакет? Сможет ли новая задача вызывать нативные инструменты, или реальный пользователь на втором компьютере захочет воспользоваться им снова?

Цель должна быть сформулирована как изменение состояния, а не как название проекта с добавлением слова «завершено».

Неизвестно, что важнее чего

Если «скорость запуска, конфиденциальность, приоритет локальных данных, лаконичность страниц, совместимость со старыми данными» фигурируют одновременно, но без указания приоритетов, то в случае конфликта Codex будет вынужден самостоятельно выбирать.

Ограничения — это не список пожеланий. Настоящие ограничения должны позволять отсеивать варианты.

Непонятно, когда остановиться

«Продолжать оптимизацию» не имеет конечной точки. Codex может постоянно находить новые возможности для рефакторинга, тестирования, создания страниц и сопутствующих функций.

Задача без условия остановки не станет завершённой сама по себе, а будет только расширяться.

Сначала опишите результат, а потом — функциональность

Перед тем как приступить к работе, попробуйте сформулировать следующее предложение:

Для [конкретного пользователя] в [конкретной ситуации] преобразовать [текущее состояние] в [целевое состояние]; мы оцениваем успех на основе [наблюдаемых доказательств], не нарушая при этом [ключевых границ].

Возьмём в качестве примера сайт с инструментами:

для пользователей инструментов для работы с изображениями и PDF-файлами, перешедших на сайт из поисковой системы, преобразовать «ситуацию, когда пользователь сталкивается с большим количеством похожих страниц с неясной ценностью» в «возможность найти несколько уникальных страниц с инструментами, позволяющими выполнять реальную работу»; оценивать успех по состоянию индексации, фактическому рабочему процессу и данным поиска после запуска, не добавляя при этом партиями страниц, не имеющих самостоятельной ценности.

Эта формулировка ещё не является полным соглашением о задаче, но уже позволяет предотвратить одну ошибку: массовое добавление страниц без реальных доказательств.

Рассмотрим пример с программным обеспечением:

Для опытных пользователей инструмента для создания

скриншотов из системного трея Windows: превратить ситуацию «приложение иногда не запускается или открывается с ошибкой» в «существующая основная рабочая среда стабильно запускается и выполняет запросы»; проверять и принимать результат с помощью непрерывного выполнения с указанием точного пути установки, журнала событий и обработки остаточных процессов, при этом не добавляя второй поисковый продукт.

Как только изменяется итоговое утверждение, изменяется и программный код.

Пусть Codex сначала продемонстрирует своё понимание

Когда цель неясна, самым дорогостоящим подходом является то, чтобы Codex реализовывал решение, полагаясь на догадки. Лучше всего перед написанием кода потребовать от него вывести четыре элемента:

- какой результат, по его мнению, действительно нужен пользователю;

- какие прокси-показатели он будет использовать;

- какие неопределённости могут привести к изменению решения;

- какой минимальный результат он готов предоставить в первую очередь для проверки правильности выбранного направления.

Вот готовое к использованию руководство по началу работы:

Пока не приступайте к реализации. Переформулируйте мои требования в виде «изменения состояния пользователя», не заменяя конечный результат количеством страниц, функций, объемом кода или количеством тестов.

Пожалуйста, укажите: 1. Целевой пользователь, сценарий использования, текущее состояние, целевое состояние; 2. Показатели, которые вы планируете использовать для оценки прогресса; 3. какие из них являются лишь прокси-показа-

телями и не могут самостоятельно служить доказательством успеха; 4. Не более 5 неизвестных факторов, которые могут существенно изменить проект; 5. Минимальный срез, позволяющий как можно раньше проверить правильность выбранного направления; 6. Четкие условия прекращения проекта.

Только после того, как я подтверждаю повторение целей, приступайте к разработке плана реализации.

Ценность этого совета заключается не в том, чтобы Codex угадал все ваши мысли, а в том, чтобы его предположения стали видимыми при минимальных затратах.

Ясно ли сформулирована цель: самопроверка с помощью шести вопросов



☐ Я описал изменения в состоянии пользователя или бизнеса, а не просто перечислил функции;



☐ Я описал целевых пользователей и конкретные сценарии использования;



☐ Я могу указать хотя бы один прокси-показатель, который может ввести проект в заблуждение;



☐ Я описал наиболее важные ограничения на случай конфликтов;



☐ Я знаю, какой минимальный результат нужно реализовать в первую очередь, чтобы определить направление;



☐ Я описал, когда следует приостановить работу, задать вопросы или отказаться от дальнейшего расширения.

Если вы не можете ответить на два из этих вопросов, не просите Codex «продолжать оптимизацию».

Сложность — это не страшно. Сложные задачи можно разбить на файлы, шаги, роли и проверки. Настоящая опасность заключается в том, что нечетко сформулированная цель маскируется под огромный объем выполняемой работы.

Codex может помочь вам двигаться быстро. Но сначала нужно решить, в каком направлении стоит двигаться быстро.

Основания и ограничения данной главы



Цифры и состояние инструментов, веб-сайтов и примеров взяты из реального проекта PixCloak, реализованного лично автором, а также из соответствующей документации (C02) и матрицы взаимодействия примеров; данные относятся только к соответствующему временному периоду и не отражают текущее состояние онлайн.



Примеры ClipDock, плагинов, платформ и интернет-магазинов взяты из общих моделей шести примеров, приведенных в этой книге; полные доказательства приводятся в последующих главах.



Задания, приведенные в тексте, представляют собой учебные версии, реконструированные автором на основе данных примеров, и не являются дословными цитатами из диалогов.

Глава 2. Он написал правильный код, почему же он всё равно поступил неправильно

Суть этой главы: Codex может блестяще выполнить задачу, которая была неправильно сформулирована. Главная обязанность человека — не писать более длинные инструкции, а сначала определить, какой результат стоит того, чтобы его достичь.

Неправильный ответ, получивший «12 баллов из 12»

Первым испытанием в проекте была задача с трёхзначным паролем.

Ранние версии выглядели вполне завершёнными: на экране отображалось 12 возможных ответов, было семь шагов разбора, подсказки о направлениях, в которых ответ был неверным, оценка в звездах, а также набор интерактивных элементов, требующих от игрока продемонстрировать процесс вывода. Внутренний набор доказательств проверялся по пунктам, и в итоге выдавался результат «12/12 PASS».

Если смотреть только на реализацию, в этом результате трудно найти недостатки. Функции реализованы, процесс прошёл, все пункты проверки выполнены. Codex не пошел на упрощения, можно даже сказать, что они выполнили тогдашние требования весьма добросовестно.

Однако, увидев реальные кадры, руководитель проекта пришёл к совершенно противоположному выводу: это не та игра, которую он хотел создать.

Перед игроком стояла не загадка, понятная с первого взгляда и позволяющая сразу же приступить к размышлениям, а «рабочий лист для рассуждений», заполнение которого сначала нужно было освоить. Так называемая защита от перебора вариантов сводилась лишь к тому, что на экра-

не отображались 12 вариантов, а игроку приходилось выполнять несколько дополнительных действий по доказательству. То, что игрок действительно должен был делать, было окружено списком вариантов, уровнем анализа, узлами доказательства и звездной шкалой, в результате чего исчезло само суть удовольствия от игры.

Самое примечательное не в том, что эта версия провалилась, а в том, что она когда-то прошла с максимальным баллом.

Это обнажило самую опасную категорию проблем при использовании Codex: **код правильный, а задача неверная; приемка пройдена, а продукт провалился.**

Codex не требует написания явных багов, чтобы сбить проект с курса. Достаточно, чтобы поставленные людьми цели, ограничения или критерии приемки были неверными, и он может мчаться в неправильном направлении на полной скорости, оставляя после себя аккуратный код, тесты и документацию, которые делают ошибку особенно правдоподобной.

«Сделать правильно» на самом деле имеет три уровня

Многие воспринимают вопрос «Сделал ли Codex всё правильно?» как единый вопрос. На самом деле он включает в себя как минимум три уровня.

Первый уровень — **правильность реализации**: работает ли код? Пройдены ли тесты? Есть ли явные ошибки?

Второй уровень — **правильность выполнения задачи**: соответствует ли результат требованиям данной задачи? Было ли действительно удалено то, что требовалось удалить? Охвачены ли все указанные страницы, платформы и границы?

Третий уровень — **правильность с точки зрения ценности**: стоит ли выполнять эту задачу? Даже если все требования выполнены, действительно ли результат решает проблему пользователя?

Взаимосвязь между этими тремя уровнями можно сформулировать следующим образом:

Правильность реализации не означает правильность задачи; правильность задачи не означает правильность ценности.

Ранние версии «Парольного уровня» неплохо справлялись с первым уровнем и, возможно, соответствовали тогдашним локальным спецификациям на втором уровне, но не прошли третий уровень: игрокам нужна была чёткая и

увлекательная головоломка, а не набор форм для подтверждения собственного мыслительного процесса.

Именно поэтому, согласно принципу, даже если «Codex сначала составляет план», этого всё равно может оказаться недостаточно. Если план строится вокруг неверной цели, то более подробный план лишь сделает эту ошибку более организованной.

Почему Codex так серьёзно ошибается

Просто свести проблему к тому, что «ИИ не разбирается в продукте», не приносит особой пользы. Более конструктивный вопрос заключается в том: чего же не хватило в этом сотрудничестве между людьми и Codex?

1. Люди задавали функциональные цели, но не задавали цели, связанные с игровым опытом

«Увеличить количество этапов рассуждений», «предотвратить беспорядочные нажатия игроков», «сделать уровни более целостными» — всё это похоже на цели, но на самом деле они описывают средства, а не конечный опыт, который получает игрок.

Если же настоящая цель заключается в том, чтобы:

- игрок в течение десяти секунд понимает, что ему нужно делать;
 - процесс рассуждений происходит в основном в голове;
 - экран служит лишь для предоставления подсказок и получения окончательного ответа;
 - сложность обусловлена самой задачей, а не операционной нагрузкой;
 - то «добавление контрольных точек» скорее всего с самого начала не должно было входить в план.
- Codex может оптимизировать записанные цели, но не спо-

собен стабильно выявлять те цели, о которых не было сказано вслух. Фраза «Я хочу создать простую, но глубокую мини-игру», возникшая в голове человека, если она не разбита на наблюдаемые критерии пользовательского опыта, в процессе реализации легко заменяется такими прокси-показателями, как «функциональная полнота», «насыщенность страниц» и «строгость игрового процесса».

2. Локальные документы стали высшим авторитетом в вопросах ошибок

В проекте уже имеются старые спецификации уровней, Harness и пакет доказательств. Они сообщают Codex, что такое «полнота» и что такое «прохождение». Таким образом, у Codex есть все основания продолжать дополнять процессы выбора кандидатов, проверки и анализа.

Проблема заключается не в том, что Codex не учитывает контекст, а именно в том, что он слишком внимательно учитывает **контекст не того уровня**.

В одном репозитории могут одновременно присутствовать принципы продукта, старые требования, тесты, исторические решения, рыночные рекомендации и временные записи. Они не являются равнозначными по своей природе. Когда порядок авторитетности не прописан чётко, наиболее конкретные и легко реализуемые локальные файлы часто перевешивают расплывчатые намерения продукта.

Поэтому «передача всех данных в Codex» не равнозначна предоставлению эффективного контекста. Эффективный контекст также должен давать ответ на вопрос:

если эти данные противоречат друг другу, чье слово будет решающим?

3. Критерии приемки измеряют то, что можно измерить, но не измеряют то, что важно

Отображается ли кандидат, завершена ли аналитика, появляется ли звезда — всё это легко проверить автоматически. А вот «сможет ли игрок быстро понять», «есть ли на экране чёткий фокус» и «не прерывается ли процесс рассуждений интерфейсом» — автоматизировать гораздо сложнее.

В результате команда легко допускает естественную ошибку: принимает показатели, которые можно измерить автоматически, за действительно важные цели.

12/12 PASS подтверждает, что двенадцать пунктов из старого набора доказательств верны, но не доказывает, что игровой опыт устроен правильно. Тестирование никогда не обладает интеллектом, превосходящим интеллект того, кто его определил.

4. Люди слишком поздно видят реальные результаты

Если руководитель впервые просматривает скриншоты только после того, как уровень написан, документация заполнена и пакет доказательств одобрен, ошибки в направлении уже привели к накоплению затрат на доработку.

В сложных проектах не следует спрашивать только в конце: «Готово ли это?», а нужно как можно раньше задать вопрос: «Это то направление, в котором мы должны двигаться?». Игравельный первый экран часто лучше, чем десятистраничный план, выявляет отклонения в понимании.

По-настоящему эффективное исправление отклонений — это не просто сказать: «Сделайте проще»

Столкнувшись с неверной версией, люди чаще всего дают такую обратную связь:

«Слишком сложно, сделайте проще. Сделайте интерфейс красивее, игровой процесс — интереснее».

Такая обратная связь выражает недовольство, но не устанавливает новых критериев оценки. Следующим шагом Codex может стать просто удаление двух панелей, сокращение нескольких абзацев текста, смена цветовой гаммы — и в результате снова будет сдана «более лаконичная сложная версия».

Отзыв, который действительно меняет направление проекта, должен преобразовывать интуицию человека в выполнимое соглашение.

Приведённое ниже описание задачи было реконструировано на основе проектной документации и журнала изменений; это не дословная цитата из разговора:

Приостановить расширение уровня. Сначала переделать первый уровень и сделать его стандартом игрового опыта для последующих уровней. Со стороны игрока оставить только три необходимых подсказки, три поля для ввода кода,

цифровую клавиатуру, подсказку, кнопку сброса и кнопку отправки. Рассуждения игрока происходят в его голове, а система проверяет только окончательный ответ. Защита от перебора не должна полагаться на увеличение количества шагов проверки. Со стороны системы необходимо доказать, что пространство допустимых ответов достаточно велико, что при наличии текущей подсказки существует только один правильный ответ, а ошибочная отправка не раскрывает информацию о частичном совпадении. Удалить полосу кандидатов, узлы доказательств, выбор обоснований, связь доказательств, задачи с принудительным переходом и штрафы в виде звезд. Задача должна выполняться на одном экране в вертикальной ориентации, без прокрутки. Сначала обновите правила продукта на самом высоком уровне, а затем измените данные, код, тесты и скриншоты первого уровня. Только после ручной проверки изображений первого уровня можно приступать к расширению остальных уровней.

Эта инструкция эффективна не потому, что она длиннее, а потому, что в ней выполнены пять ключевых действий.

Во-первых, она **останавливает привычку допускать ошибки**. Речь идет не о том, чтобы одновременно расширять старую версию и заниматься косметическими доработками, а о том, чтобы четко «приостановить дальнейшее расширение».

Во-вторых, **преобразование оценок пользовательского опыта в структурные ограничения**. «Простота»

больше не является просто прилагательным, а определяет, какие элементы разрешено оставлять на экране, а какие — запрещено.

В-третьих, разделяем задачи игроков и проверку со стороны разработчиков. Доказательство уникальности по-прежнему важно, но его должны обеспечивать перечислители и тесты со стороны разработчиков, а не перекладывать на игроков.

В-четвертых, сначала нужно изменить высший авторитет, а уже потом — код. В противном случае старые документы в следующем задании снова вернут Codex в исходное русло.

В-пятых, установите условия продолжения работы. Не «после завершения изменений автоматически добавить тридцать уровней», а сначала предоставить минимальный срез, чтобы можно было подтвердить правильный вектор развития.

Если сопоставить задания до и после исправления отклонений, разница станет более очевидной:

Аспект

Задачи, которые легко сбивают проект с курса

Задачи после исправления

Цель

Усовершенствовать процесс рассуждений, чтобы игроки

не нажимали наугад

Создать головоломку, которая позволит новичкам быстро приступить к мыслительному процессу и отправить только окончательный ответ

Методы

Добавление вариантов ответов, узлов доказательств, выбора обоснований и анализа

Расширение пространства возможных ответов, общее оценивание, неразглашение частичной информации в случае неудачи

Авторитет

Продолжать следовать старым правилам Harness и старым наборам доказательств

Сначала обновляются правила самого верхнего уровня; старые спецификации теряют силу в случае конфликта с ними

Область применения

Одновременно с доработкой первого уровня продолжать расширение

Приостановить расширение, завершить только продольное разделение первого этапа и соглашение о совместном использовании

Приемка

Полный набор функций, все старые пункты проверки отмечены зеленым

Уникальность математических вычислений, правильное взаимодействие, скриншот на одном экране прошел проверку и подтверждён человеком Направление развития

Условия остановки

Четких условий нет, после завершения игра продолжается автоматически

Не допускается копирование на остальные уровни до ручного подтверждения

Слева — не просто «плохая подсказка», справа — не какая-то волшебная формулировка. Настоящее изменение заключается в том, что человек начинает нести ответственность за цели, авторитет, границы и доказательства.

От простого требования к договору о выполнении задания

В руководстве по использованию Codex от OpenAI рекомендуется в более важных заданиях указывать цели, контекст, ограничения и условия завершения; сложные или неоднозначные задачи можно сначала спланировать или позволить Codex с помощью вопросов конкретизировать расплывчатые идеи. Это отличная отправная точка. [1]

Однако в реальных проектах обычно требуются ещё три дополнительных элемента: текущие факты и неизвестные переменные, полномочия и риски, а также условия прекращения. В совокупности они составляют «Семь элементов задания Codex», которые проходят красной нитью через всю книгу.

1. Целевой результат

Не ограничивайтесь описанием «какую функцию нужно реализовать», а укажите также «кому предназначен результат, какую проблему он решает и что получает пользователь после выполнения».

Цель данного примера — не «переработать интерфейс первого уровня», а:

создать одноэкранный уровень с кодом, в котором новичок сможет быстро начать игру, где логические выводы будут происходить в голове, а система будет принимать только окончательный ответ.

2. Текущие факты и неизвестные

Факты должны быть взяты из существующего кода, данных, скриншотов или результатов выполнения; неизвестную информацию следует четко обозначить как неизвестную, а не заполнять догадками.

Факты на момент включают: в старой версии было 12 вариантов и семь шагов анализа, внутренний набор доказательств показал результат 12/12 PASS, а фактический скриншот был отклонен руководителем проекта.

Среди неизвестных факторов: сможет ли начинающий игрок действительно понять правила за 10 секунд, будут ли действия на реальном сенсорном устройстве выглядеть естественно. Эти вопросы нельзя считать решенными на основании результатов автоматического тестирования.

3. Авторитетный контекст

Перечислите материалы Codex, которые необходимо прочитать, и укажите порядок разрешения конфликтов. Например:

Текущие `PRODUCT_RULES` имеют приоритет над старыми Harness; текущий код и результаты выполнения имеют приоритет над устаревшими скриншотами; рыночные рекомендации не должны перекрывать подтвержденные принципы продукта.

Контекст не должен быть чрезмерно обширным. Главное — дать Codex понять, какие данные могут изменить вывод, а какие служат лишь исторической справкой.

4. Объем и нецелевые задачи

Описание области применения указывает, что необходимо изменить в данном случае; описание нецелей указывает, какие заманчивые сопутствующие задачи не следует выполнять «заодно».

На данный момент мы переделываем только первый уровень и его общие условия; не будем одновременно добавлять рейтинги, доску заданий, социальную систему или дополнительные уровни. Если не указать «нецели», чем способнее Codex, тем больше вероятность появления лишней работы.

5. Полномочия и риски

Какие операции можно выполнять автоматически, а какие должны подтверждаться человеком?

Можно разрешить изменение исходного кода, тестирование и документирование; решение об общем направлении игрового процесса требует подтверждения руководителя; реальный релиз, управление учетными записями и права на коммерческие материалы не могут считаться завершенными только на основании успешного прохождения локального тестирования.

6. Этапы выполнения и контрольные точки

Сложные задачи не следует выполнять по принципу «сначала написать весь код, а потом провести единую приемку». Следует сдавать минимальные, но полные фрагменты:

Правила продукта → Данные первого уровня → Играбельная страница → Автономное решение → Тестирование взаимодействия → Скриншоты → Ручное подтверждение на-
правления.

Данный фрагмент одновременно включает в себя замысел, реализацию и подтверждающие доказательства. Любая ошибка на любом уровне может быть обнаружена до того, как фрагмент будет скопирован на тридцатый уровень.

7. Доказательства приемки и условия остановки

«Выглядит как готовое» — это не приемка. Приемка должна сопровождаться конкретными доказательствами: перечислением всех возможных решений, проверкой единственности решения, поведением при отправке ошибочного ответа, сбросом состояния, скриншотом целевого разрешения, а также результатами первого слепого тестирования игроками, которые все еще подлежат проверке.

Условия остановки не менее важны: если в правилах продукта есть противоречия, не удалось доказать уникальность, по-прежнему требуется прокрутка первого экрана или нажатие ещё не подтверждено человеком, Codex должен прекратить расширение и подать отчёт, а не продолжать работу, пойдя на компромисс.

Когда исправлять, а когда начинать с нуля

После внедрения Codex скорость изменения кода увеличивается, и люди, напротив, легче попадают в ловушку упущенных затрат: раз уж было проведено столько тестов и подготовлено столько документации, не выгоднее ли просто немного доработать старый вариант?

Критерием принятия решения должно быть не «сколько кода уже написано», а «на каком уровне произошла ошибка».

Если расстояние между кнопками неверное, текст неясен или какой-то статус не сброшен, это отклонение на уровне реализации, и локальное исправление обычно оправдано.

Если страница по-прежнему выполняет нужные пользователю задачи, но процесс протекает негладко, это отклонение на уровне задач, и можно перераспределить процессы и компоненты.

Если игрок изначально не должен был выполнять эти действия, то ошибка произошла на уровне ценности. В этом случае дальнейшее исправление ошибок лишь будет поддерживать структуру, которая не должна существовать.

Уровень с паролем в конечном итоге не стал «рабочей средой для оптимизации доказательств», а был преобразован путём удаления полосы кандидатов, цепочки доказа-

тельств, выбора обоснований, связей между доказательствами и штрафных звёздочек, в результате чего страница была сведена к трём записям, механическому парольному замку и цифровой клавиатуре. Соответствующие изменения затронули 31 файл: было добавлено 522 строки и удалено 903 строки.

То, что удалено больше, чем добавлено, не означает, что объем работы был небольшим. Это свидетельствует о том, что команда наконец-то перенесла сложность с игроков обратно на сторону разработчиков: игроки просто отправляют ответы, а система разработчиков отвечает за проверку правильности вопросов.

Что должно подтверждать автоматическое тестирование, а что — нет

Первый уровень после переработки имеет более строгие автоматические проверки: система перебирает 720 комбинаций из трёх разных цифр и проверяет, что текущая подсказка содержит только один объявленный ответ; при ошибочной отправке возвращается только общий статус «неудача», без указания, какая цифра верна; при перезапуске ввод очищается; страница спроектирована как единый экран размером 720×1280; при выполнении обёртка не только проверяет код завершения, но и сканирует ошибки выполнения и метку «PASS».

Эти доказательства чрезвычайно важны. Без них «лаконичность» могла бы просто скрыть ошибки ещё глубже.

Однако они по-прежнему не могут доказать:

- сможет ли человек, впервые столкнувшийся с игрой, быстро понять правила;

- станет ли механический сейф действительно центром визуального внимания;

-

возникает ли у игрока при разгадке ощущение «а, вот в чём дело»;



насколько естественны сенсорные управления на реальных устройствах WeChat;



готовы ли данные по проверке платформы, обратным звонкам по рекламе и соблюдению нормативных требований.

Хорошая приемка — это не просто пометка «PASS» во всех результатах, а допущение трёх вариантов честной оценки:



PASS

: имеются доказательства, подтверждающие соответствие требуемому уровню;



FAIL

: имеются доказательства несоответствия;



PENDING / UNKNOWN: когда до статуса нет достаточных доказательств, и требуется дополнительная проверка вручную, на реальном устройстве или с помощью внешних систем.

«Неизвестно» — это не провал. Провал — это когда «неизвестно» маскируется под «завершено».

Эта проблема характерна не только для игровых проектов

Одна и та же ошибка может проявляться по-разному на сайте, в программном обеспечении и в операционных задачах.

На контент-сайтах задача «создать ещё сто SEO-страниц» может быть выполнена в полном объёме, но данные поисковых систем уже показывают, что большое количество старых страниц не проиндексировано. В данном случае на самом деле не хватает не скорости написания, а текущей базовой линии и доказательств того, «зачем вообще нужно увеличивать количество страниц».

В настольном программном обеспечении тестирование и сборка могут пройти успешно, но при реальном запуске все равно могут возникнуть остатки процессов, сбои при обновлении или дублирование входов. В данном случае не хватает многоуровневого определения, прослеживающего путь от исходного кода до реального пользовательского сценария.

При управлении интернет-магазином автоматизация может успешно нажать кнопку и отобразить сообщение «Операция выполнена успешно», но это не означает, что нужный товар был сохранен в правильном статусе, и тем более не означает, что было получено разрешение на выпуск. В данном случае не хватает подтверждения объекта, определе-

ния границ полномочий и бизнес-проверки результатов после выполнения операции.

На первый взгляд это проблемы SEO, программной инженерии и браузерной автоматизации; с точки зрения взаимодействия человека и машины это одна и та же проблема: **принятие успеха отдельного действия за успех конечного результата.**

Правильное распределение обязанностей между человеком и Codex

Этот пример не означает, что Codex подходит только для написания кода и не может участвовать в проектировании продукта. Напротив, он выполнил огромный объем межролевой работы: прочитал правила и старую реализацию, рефакторизовал соглашения о данных, изменил сцены в Godot, написал тесты для решения задач и проверки состояний, сгенерировал скриншоты в качестве доказательств, обновил документацию и расширил новые правила до повторяемых поперечных разрезов уровней.

Однако сильные стороны человека и Codex находятся в разных областях.

Codex преуспевает в следующем:

- выявлять затронутые области в большом количестве файлов;
- воплощение чётких правил в коде, тестах и документации;
- выполнение повторяющихся, но требующих согласованности изменений;

- генерировать поддающиеся проверке отчеты о различиях и доказательствах проверки;

- отслеживание технических последствий после обнаружения конфликтов.

Человек должен отвечать за:

- определение проблем, которые действительно стоит решать;

- определять, когда промежуточные показатели заменяют реальные цели;

- принимать решения относительно пользовательского опыта, бренда, рисков и коммерческих последствий;

- отказывать в признании задачи «выполненной» при недостаточности доказательств;

- принимать решение о продолжении, доработке или прекращении проекта.

Самый опасный способ сотрудничества — это когда человек лишь формулирует расплывчатую цель, Codex берет на себя все промежуточные решения, а в итоге человек видит только «тест пройден». Самый эффективный способ со-

трудничества — это когда человек фиксирует ключевые решения в виде соглашения, Codex воплощает это соглашение в работу, а человек принимает следующее решение на основе реальных доказательств.

Готово к использованию: трёхуровневая проверка правильности

Прежде чем поручить Codex выполнение важной задачи, ответьте на следующие девять вопросов.

Правильность ценности

Кто является конечным пользователем и что он действительно хочет сделать?

Из-за чего функция может оказаться бесполезной, даже если она реализована в полном объеме?

Не принимаем ли мы какой-то легко измеримый показатель за истинную цель?

Правильность задачи

Что необходимо изменить в данном случае, и что точно не следует менять?

В случае противоречия данных, какой файл, какое правило или какое лицо обладает правом окончательного толкования?

Какой минимальный выполнимый результат позволит как можно раньше определить направление?

Правильная реализация

Какие автоматические проверки могут подтвердить корректность кода и поведения?

Какие результаты необходимо проверить с помощью скриншотов, реальных устройств, первых пользователей или внешних систем?

Какие неизвестные элементы необходимо оставить в статусе PENDING / UNKNOWN, не объявляя их завершёнными?

Если на три или более из этих девяти вопросов вы не можете ответить, лучшим следующим шагом, как правило, будет не то, чтобы Codex сразу приступил к написанию кода, а то, чтобы он сначала взял у вас интервью, изучил авторитетные источники, повторил полученную информацию своими словами и составил договор о задании, подлежащий подтверждению.

Совет по началу работы, который можно скопировать напрямую

Пока не реализуйте. Пожалуйста, сформулируйте мои идеи в виде соглашения о задании и выявите в нём пробелы, которые могут привести к ситуации, когда «код написан правильно, но результат неверный».

Пожалуйста, выведите по порядку: 1. Целевые пользователи и ожидаемые результаты; 2. Известные факты, допущения и неизвестные величины; 3. Авторитетные источники, которые необходимо изучить, и порядок их приоритета; 4. Объем данной задачи и четко обозначенные нецели; 5. Действия, которые вы можете выполнить, и решения, которые должны быть утверждены мной; 6. Минимальные поперечные срезы и контрольные точки ручной проверки; 7. Иерархические доказательства приемки, пункты в статусе «PENDING» и условия остановки.

Затем в не более чем 200 словах изложите своё понимание задачи и перечислите не более 5 вопросов, которые могут существенно изменить план. Приступайте к реализации только после моего подтверждения.

Эти указания не гарантируют, что Codex никогда не будет допускать ошибок. Никакие указания не могут этого обеспечить.

Его истинная ценность заключается в том, чтобы ошибки

выявлялись раньше, их последствия были меньше, доказательства легче проверялись, а результаты легче отменялись, а также в том, чтобы люди перестали ошибочно считать проект успешным из-за того, что «Codex очень занят», «было изменено много файлов» или «тесты показали зеленый цвет».

Хорошее проектирование задач заключается не в искоренении ошибок, а в том, чтобы не дать им незаметно разрастаться.

Основания и ограничения данной главы



Факты, приведенные в игровых примерах, взяты из документации по проектам, правил продукта, доказательств QA, статистики коммитов и записей верификации. Перед публикацией части, содержащие оригинальные высказывания руководителей, подлежат подтверждению самими авторами.



Указания в тексте «восстановлено на основе данных проекта и журналов изменений» предназначены для воспроизведения в учебных целях и не являются дословными цитатами из диалогов.



[1] Официальное руководство OpenAI Codex по лучшим практикам и подсказкам: рекомендуется четко определять для важных задач «Цель» (Goal), «Контекст» (Context), «Ограничения» (Constraints) и «Когда задача выполнена» (Done when), а в случае сложных или неоднозначных задач сначала составлять план или прояснять детали. См.

[«Codex Best Practices»](#)

и

[«Prompting»](#)

».



Основной источник примеров: архив проекта автора «Игры ума: черновики примеров Codex».

Глава 3. Codex — исполнитель, человек — руководитель

Суть этой главы: передача работы Codex не означает передачу целей, полномочий и ответственности за последствия. Человек отвечает за решение, что стоит делать, что можно делать и когда принимать решения; Codex отвечает за выполнение, сбор данных и отчетность в рамках установленных границ.

Вход в админ-панель означает, что можно приступать к работе?

В одном из реальных примеров управления интернет-магазином Codex должен был помогать в проведении исследований, выборе товаров, ценообразовании и подготовке черновых версий товаров. В браузере уже был выполнен вход в админ-панель Jingmai, и все кнопки на странице были активны.

С точки зрения функциональных возможностей, он «может выполнять операции».

Однако при просмотре в режиме «только для чтения» из-за отложенной загрузки страницы произошло смещение карточек, и рядом с объектом, который планировалось просмотреть, появилась кнопка «Разместить товар одним нажатием». Если автоматизированные операции будут по-прежнему зависеть от положения элементов на экране, это может привести к тому, что просмотр будет принят за реальное бизнес-действие.

В других случаях на странице появляется всплывающее окно «Товар успешно добавлен», но состояние в последующем списке и в деталях не совпадает. Уведомление об успешном добавлении подтверждает, что браузер получил обратную связь, но не подтверждает, что нужный товар с правильными полями перешел в правильное бизнес-состоя-

ние.

Эти две проблемы указывают на одну и ту же границу: доступ не означает наличие полномочий; возможность нажать не означает, что нужно нажимать; успешное нажатие не означает завершение бизнес-операции.

Если человек просто скажет: «Станьте управляющим и запустите работу магазина», Codex не сможет автоматически понять, какие решения можно принимать от имени владельца, какие действия приведут к необратимым последствиям, а за какие цены, товары или статус выпуска ответственность должен нести владелец магазина.

Так называемая «роль Codex в качестве управляющего» более точно следует понимать следующим образом: Codex берет на себя работу управляющего по подготовке, анализу, координации выполнения и систематизации доказательств, но не заменяет владельца магазина в плане его юридического статуса, ответственности за учетную запись и принятия окончательных управленческих решений.

Управляющий — не самый занятой человек

Некоторые пользователи, как только осознают, что ИИ может ошибаться, уходят в другую крайность: на каждом шагу указывают Codex, какой файл открыть, какую строку изменить, на какую точку нажать, какую команду выполнить.

На первый взгляд это кажется безопасным, но на самом деле человек превращается в медленный пульт дистанционного управления. Codex лишается возможности искать, сравнивать, разбирать и проверять, а у человека нет сил постоянно контролировать каждое мельчайшее действие.

Настоящий руководитель не должен лично принимать решения по каждому фрагменту кода. Он отвечает за пять задач более высокого уровня:

Определение результата: зачем это делается, для кого предназначено, при каких условиях результат будет иметь ценность;

Установка границ: какие принципы, бюджет, полномочия и риски нельзя превышать;

Выбор доказательств: что может подтвердить завершение, а что — лишь частичный успех;

Установка контрольных точек: какие компромиссы должны быть подтверждены человеком, прежде чем можно будет продолжить;

Принятие решений: кто несет ответственность за релиз, сделки, соблюдение нормативных требований, а также за влияние на бренд и пользователей.

Если эти пять пунктов четко определены, Codex может обладать значительной автономией в реализации проекта.

Разделение полномочий на три категории

Распределение обязанностей между человеком и машиной чаще всего приводит к ошибкам, поскольку термин «задача» смешивает три совершенно разных вида полномочий.

Право на внесение предложений

Codex может искать данные, читать репозитории, выявлять проблемы, сравнивать варианты и вносить предложения. Предложения не вносят прямых изменений во внешний мир, поэтому обычно им можно предоставить большую степень автономии.

Например, он может предложить три модели ценообразования, указав валовую прибыль, риск возврата товаров и пробелы в доказательствах для каждой из них.

Право на исполнение

Codex может в пределах четко очерченных границ изменять файлы, проводить тестирование, создавать черновики или заполнять непубликуемый контент. Право на исполнение должно быть привязано к объекту, области действия и обратимости.

Например, он может создавать до N черновиков товаров, не выставленных на продажу, но при этом должен привязываться к точным SKU и после завершения каждого черновика возвращаться к списку и деталям.

Право принятия решений

Действия, связанные с внешними публикациями, платежами, возмещениями, рекламным бюджетом, коммуникацией с клиентами, юридическими обязательствами, производственными данными и рисками для бренда, обычно требуют явного одобрения со стороны человека.

Право принятия решений нельзя выводить из «статуса входа в систему», «доступности инструмента» или «прежнего разрешения на аналогичные действия».

После разделения этих трех видов полномочий многие неоднозначные полномочия исчезнут сами собой. Codex может обладать широкими полномочиями по выдвижению предложений и контролируемыми полномочиями по исполнению, но не обладает автоматически правом принятия окончательного решения.

Карта полномочий, более полезная, чем «полная автоматизация»

Перед началом проекта разделите действия на три зоны:

Зона

Поведение по умолчанию

Типичные действия

Зона самостоятельности

Codex — можно выполнять напрямую и сохранять доказательства

Чтение файлов, анализ данных, изменение исходного кода, выполнение тестов, подготовка черновиков

Зона подтверждения

Предложение решения или карточки утверждения, ожидание подтверждения

Масштабные удаления и изменения, изменение направления развития продукта, выполнение видимых действий с использованием реальных учетных записей

Запрещённая зона

Данное задание не подлежит выполнению

Официальный релиз, оплата, возврат средств, размеще-

ние рекламы, отправка сообщений клиентам, удаление производственных данных

Границы зон не являются постоянными. Разрешение на выпуск определенной тестовой версии в рамках одного задания не означает, что все последующие выпуски будут автоматически разрешены; разрешение на создание черновиков также не означает разрешение на нажатие кнопки «Опубликовать».

Разрешение должно быть привязано к четырём элементам:

Действие + Объект + Область применения + Срок действия.

Формулировка «можно изменять цену» недостаточно чётка; «в рамках данного задания можно рассчитать рекомендуемую цену для 5 черновиков из списка, которые ещё не выставлены на продажу, но нельзя подавать официальную цену» — вот и есть допустимые границы.

Не следует дистанционно контролировать шаги, а нужно управлять соглашением

Не нужно указывать Codex, как действовать на каждом этапе, но следует требовать, чтобы он предоставлял одно-значные результаты на ключевых этапах.

Возьмём в качестве примера подготовку черновика интернет-магазина. Неэффективный способ:

открыть админ-панель, щелкнуть второе меню слева, найти третью вкладку, затем нажать кнопку в правом нижнем углу...

Как только страница меняется, эта система координатных инструкций теряет силу, и человек берет на себя всю ответственность за планирование действий.

Лучшее задание:

подготовить не более 5 черновиков товаров из подтвержденного списка, которые еще не выставлены на продажу. Перед каждой операцией проверять SKU, название и источник поставки; запрещается публиковать, изменять рекламу или связываться с клиентами. По завершении одновременно перепроверять количество черновиков, элементы списка и поля с подробностями; при несоответствии этих трех показателей пометить задачу как «UNKNOWN» и остановить

процесс. В конце сформировать пакет документов для утверждения владельцем магазина.

В этом задании не прописаны все отдельные действия, но контролируются цели, объекты, полномочия, подтверждающая документация и обработка исключений. Codex может выбирать путь в зависимости от фактической ситуации на странице, а человек вмешивается только там, где действительно требуется принятие решения.

Четыре категории решений, которые нельзя передать на аутсорсинг

Оценка ценности

Стоит ли реализовывать ту или иную функцию, имеет ли страница самостоятельную ценность, интересна ли игра — в конечном итоге все сводится к реальным пользователям и бизнес-целям. Codex может предоставлять доказательства и предложения, но не может принимать решения о выборе продукта вместо человека.

Принятие рисков

Разрешать ли использование реальных учетных записей, можно ли изменять производственные данные, следует ли принимать тот или иной правовой или безрепутационный риск — все это должны решать лица, уполномоченные нести ответственность за последствия.

Достаточность доказательств

Являются ли «зеленые» результаты тестирования, нормальные скриншоты и появление всплывающего окна об успешном завершении достаточными доказательствами для признания задачи «выполненной» — это зависит от уровня реальной реализации задачи. Необходимо отклонять заявления о завершении, если уровень доказательств не соответствует уровню реализации.

Компромиссы

Скорость, стоимость, качество, совместимость и удобство обслуживания часто вступают в противоречие. Codex может перечислить возможные издержки, но без соответствующих полномочий не должен выбирать от имени проекта, чем пожертвовать.

Какие обязанности Codex должен брать на себя по собственной инициативе

Подчеркивание ответственности человека не означает превращение Codex в простого машиниста, ожидающего приказов. Хорошо спроектированная задача должна требовать от него проактивного выполнения следующих действий:

- перед выполнением повторять цели и полномочия;
- изучать авторитетные источники и указывать на противоречия;
- разделять факты, предположения и неизвестные моменты;
- предлагать минимальные вертикальные изменения;
- самостоятельно вносить изменения и проводить проверку в рамках установленных границ;
- останавливаться при обнаружении действий с высоким риском или необратимых действий;



сообщать о фактических доказательствах, а не только о выполненных шагах;



проводить анализ повторяющихся ошибок и предлагать закрепление долгосрочных правил.

Люди управляют направлением и ответственностью, а Codex — сложностью выполнения. Ни те, ни другие не должны сводиться к построчному дистанционному управлению.

Можно использовать напрямую: карта полномочий ответственного лица

Цель задачи: [Кто, в каких условиях и какой результат должен быть достигнут]

Codex может самостоятельно: - [Просмотр в режиме «только для чтения»] - [Локальные, восстанавливаемые изменения] - [Проверка разрешений на выполнение]

Сначала необходимо убедиться: - [Решения, которые изменяют направление развития продукта] - [широкомасштабные или трудновосстановимые изменения] - [Действия, которые повлияют на работу реальных пользователей или внешних систем]

В данном случае запрещено: - [релизы, платежи, удаления, отправки, производственные операции и т. д.]

Объекты: [конкретные хранилища, файлы, SKU, среды, учетные записи или версии]

Точки ручной проверки: [по каким доказательствам и кто принимает решение о продолжении]

Правила обнаружения аномалий: в случае неуникальности объекта, несоответствия при обратном считывании состояния, неясных прав доступа или появления непредвиденных внешних воздействий немедленно остановите процесс,

поставьте состояние как UNKNOWN, не делайте предположений и не повторяйте попытку вслепую.

Самопроверка ответственного лица



[] Определил ли я результат, а не просто распределил задачи;



[] Разграничил ли я права на внесение предложений, исполнение и принятие решений;



[] Связаны ли полномочия с четко определенными действиями, объектами, сферой действия и сроками;



[] Предусмотрены ли ручные контрольные точки для действий с высоким риском;



[] Может ли Codex самостоятельно выбирать путь реализации в пределах безопасных границ;



[] Существуют ли правила остановки и «UNKNOWN» в случае аномалий;



[] Видел ли человек, принимающий окончательное решение, доказательства соответствующего уровня.

«Руководитель» — это не ролевой термин, позволяющий ИИ действовать по своему усмотрению. Это система ответ-

ственности.

Codex может брать на себя значительный объём работы и выполнять повторяющиеся процессы стабильнее, чем человек. Однако решение о том, зачем существует проект, какие последствия являются приемлемыми и когда проект считается действительно завершённым, по-прежнему должно приниматься трезво мыслящим человеком.

Основания и границы данной главы



Пример интернет-магазина взят из реального проекта интернет-магазина «Jingmai», которым лично управляет автор, а также из соответствующих доказательственных материалов (С06) и матрицы взаимодействия с примером; отзывы на конкретных страницах обобщены в соответствии с состоянием бизнеса, информация об учетных записях, товарах и заказах не разглашается.



Примеры разрешений в тексте представляют собой учебные тексты, реконструированные в соответствии с методологией рассмотрения примеров, и не являются дословными цитатами из диалогов.



Реальные полномочия в отношении публикации, транзакций, рекламы, обслуживания клиентов и послепродажного обслуживания варьируются в зависимости от организации и платформы; в данной главе представлен метод проектирования задач, который не заменяет правил платформы или профессиональных рекомендаций по соблюдению нормативных требований.

Глава 4. Подсказки — это лишь отправная точка, а не полноценный метод

Суть этой главы: одно подсказывание может инициировать задачу, но само по себе не способно поддерживать долгосрочный проект. Надежное сотрудничество требует объединения целей, долгосрочных правил, плана выполнения, проверки доступа и анализа результатов в единую систему.

Почему даже очень длинное задание всё равно может сойти с курса

Одна из платформ рыночной аналитики реализовывала проект, следуя этапам от G0 до G7. Задачи охватывали архитектуру, безопасность, данные, инструменты MCP, интерфейсы, плагины, подачу материалов и тестирование. На каждом этапе были получены результаты, и в итоге все показатели стали «зелеными».

Судя по итоговому отчёту, это типичный пример того, как «хорошие указания приводят к хорошим результатам»: требования были подробными, этапы чётко определены, проверки — исчерпывающими.

Однако независимая проверка всё же выявила несколько ключевых проблем: планировщик не был должным образом подключён к Cadence, виджет отображал успешные запросы на извлечение данных как пустые результаты, в Discovery имелись конфликты между идеальными ключами, а в Preflight обнаружилось уязвимости, связанные с типами данных.

Речь идет не о полном отсутствии реализации функций, а о том, что связи между модулями, исключительные состояния и реальные бизнес-договоренности не были охвачены исходными проверками.

Что ещё важнее, в ходе реализации длительных задач фак-

тическая ситуация в проекте меняется: файлы изменяются, старые решения теряют актуальность, появляются новые риски, а сфера применения переосмысливается. Первая подсказка не может заранее учитывать все факты, которые появятся позже.

Поэтому проблема не в том, что «подсказка недостаточно длинная». Проблема в том, что постоянно меняющийся проект рассматривается как система вопросов и ответов, требующая всего одного ввода и одного вывода.

На что способен промпт

Подсказка лучше всего подходит для выполнения четырёх задач:

- описать результат, который необходимо получить в данный момент;

- указать актуальный контекст;

- указать ключевые границы и формат вывода;

- определить, что необходимо проверить до завершения текущего раунда.

В руководстве по лучшим практикам Codex от OpenAI «Goal», «Context», «Constraints» и «Done when» рассматриваются как хорошая отправная точка для важных задач; в случае сложных или неоднозначных задач рекомендуется сначала составить план или позволить Codex сначала конкретизировать идеи с помощью вопросов. [1]

Это означает, что подсказка является **точкой входа в совместную работу**, а не всей системой совместной работы.

Даже самый чёткий вход не решает три категории долгосрочных проблем:

-

- будет ли следующая задача помнить те же правила;

-

- кто имеет приоритет при конфликте нескольких файлов в

репозитории;



пропустил ли сам разработчик в своих тестах непредвиденные пути.

Почему одноразовый чат не работает

Правила существуют только в памяти человека

Люди говорили: «Приоритет локальной версии», «Не создавайте вторую версию страницы», «Не используйте тестовую среду вместо производственной», но эти решения не стали постоянными правилами проекта. После нескольких циклов задач Codex заново считывает репозиторий, и старый README или конкретная реализация могут оказаться более заметными, чем исторические высказывания людей.

План не обновляется с учётом фактических данных

Предположения, сделанные в начале планирования, могли быть опровергнуты результатами выполнения. Если по-прежнему рассматривать первоначальный план как список команд, Codex будет игнорировать сигнал «этот этап уже не стоит продолжать» ради завершения этапа.

В объявлении о завершении отсутствует возможность опровержения

Когда один и тот же исполнитель разрабатывает тесты, реализует функционал, проводит тестирование и подводит итоги завершения, ему естественно легче проверить свой собственный предполагаемый путь. Дело не в том, что он намеренно что-то скрывает, а в том, что мир тестов обычно совпадает с миром в голове реализатора.

Отсутствие накопления опыта

Если исправление отклонений ограничивается лишь разговором, в следующем задании ошибка повторится. Проект постоянно несет одни и те же затраты на коммуникацию.

Переход от подсказок к пятиуровневой системе задач

Первый уровень: подсказки для текущего раунда

Указывайте только цели, контекст, границы и условия выполнения, относящиеся к данному раунду. Не копируйте весь руководство команды в каждое сообщение.

Второй уровень: договор о задаче и план выполнения

Для сложных задач сначала составьте соглашение, подлежащее проверке: факты и неизвестные, объем и нецели, полномочия, сегменты, доказательства и условия прекращения. План — это не фиксированный маршрут, а оптимальный маршрут с учетом текущих доказательств; при изменении фактов его следует явно обновлять.

Третий уровень: постоянные правила проекта

Долгосрочная стабильная структура репозитория, команды, запреты, методы проверки и требования к рецензированию должны быть зафиксированы в файле AGENTS.md или в соглашении по проекту, а не зависеть от того, что пользователи будут каждый раз напоминать об этом.

В официальной документации файл AGENTS.md описывается как файл описания в открытом формате, ориентированный на агентов, подходящий для записи структуры репозитория, команд выполнения, инженерных соглашений, запретов, а также определения того, «что считается завершением». Более конкретные правила, относящиеся непосредственно к текущему каталогу, могут переопределять правила верхнего уровня. [1]

Постоянные правила должны быть краткими и точными. Если при каждой ошибке вставлять весь отрывок чата, он превратится в накопленную историю, в которой никто не сможет определить приоритеты.

Четвёртый уровень: исполняемые ограничения

Если важные правила можно проверить автоматически, не ограничивайтесь их текстовым описанием. Например:

- архитектурная проверка запрещает зависимости одного уровня от другого;

- проверка доступа к релизам контролирует конфиденциальные поля и содержимое пакетов;

- задачи должны выполняться в точной установке кеша, а не в репозитории разработки;

- перед внешней публикацией должна существовать запись о ручном утверждении.

Документация указывает Codex, как следует поступать, а контроль доступа затрудняет незаметное прохождение отклонений.

Пятый уровень: независимая проверка и ретроспектива

Сложные поставки требуют проверки, способной опровергнуть вывод о «завершении». Рецензент должен исходить из требований и доказательств, а не просто повторять выводы исполнителя.

При обнаружении повторяющихся проблем анализ должен дать ответ: чего не хватало — однократного напоминания, постоянного правила или автоматического контроля доступа? Только при применении исправлений на правильном уровне проект станет по-настоящему более надежным.

Где следует разместить тот или иной опыт

Не все требования нужно записывать в файл AGENTS.md, и не все привычки должны превращаться в автоматические проверки.

Информация

Подходящее место

Причина

В данном случае изменяется только страница входа

Подсказки/задания данного раунда

Объем одноразовых изменений

Выполнение определённого набора тестов перед каждой отправкой

AGENTS.md

Долгосрочные соглашения репозитория

Запрещено записывать производственные ключи в репозиторий

AGENTS.md + автоматическое сканирование

Долгосрочные правила, подходящие для автоматического выполнения

Бюджет на данный раунд не должен превышать определенную сумму

Договор на выполнение задания

Связано с текущим разрешением

Еженедельное формирование отчёта

Возможность повторного использования рабочих процессов или автоматизации

Повторяется и выполняется в фиксированные сроки

Добавление регрессионных тестов после обнаружения аналогичных ошибок

Анализ + контроль доступа

Превращение опыта в системные возможности

Практическое правило:

если это действует только в данном случае — поместите в подсказку; если это постоянно актуально для данной задачи — поместите в соглашение; если это долгосрочно актуально для репозитория — запишите в правила проекта; если это необходимо выполнять каждый раз — превратите в контроль доступа; если это повторяется периодически — только тогда рассматривайте возможность автоматизации.

Различие между «общими подсказками» и «системой задач»

Измерение

Зависит только от общих указаний

Использование системы задач

Цель

Описание в начале один раз

Возможность вернуться к целевому результату на каждом этапе

Контекст

Однократная загрузка большого объема данных

Чтение актуальных данных в порядке приоритета

Изменения

Легко рассматривается как отклонение от первоначального плана

Новые факты вызывают явное обновление плана

Долгосрочные правила

Зависимость от истории чата

Добавляются в файл AGENTS.md или в соглашения по

проекту

Проверка

Самопроверка исполнителя и подведение итогов

Многоуровневая система контроля доступа, фактического выполнения и независимой проверки

Неудача

Продолжить: добавить подсказки, переформулировать объяснение

Анализ того, к какому уровню относится ошибка, и закрепление исправлений

Система задач — это не просто набор дополнительных документов. Её цель, напротив, заключается в том, чтобы каждая информация появлялась только в нужном месте и имела чёткий приоритет.

Как запустить долгосрочную задачу

Приведенные ниже рекомендации подходят для многоэтапных проектов. Они не напишут за вас правила проекта, но помогут сначала создать правильную основу для совместной работы.

Это многоэтапная задача. Пока не приступайте к реализации, сначала создайте систему задач.

1. Читать файлы README, AGENTS.md, правила продукта и входные данные для проверки из текущего репозитория;
2. Составить список конфликтов данных и порядка авторитетности;
3. Разграничить текущие факты, историческое состояние, предположения и неизвестные данные;
4. Разбить цель на этапы, которые можно выполнять и восстанавливать независимо друг от друга;
5. Описать для каждого этапа входные данные, результаты, доказательства проверки и условия остановки;
6. Отметить, какие правила относятся только к данной задаче, а какие должны стать постоянными правилами репозитория;
7. Указать, какие выводы требуют независимой проверки и не могут быть подтверждены самим исполнителем.

Сначала предоставьте договор о выполнении задачи и план первого этапа. Только после того, как первый этап будет успешно выполнен и проверен вручную, можно переходить к следующему этапу.

Если новые данные опровергают первоначальный план, сначала обновите факты, сферу воздействия и план; не продолжайте работу только для того, чтобы завершить старый список.

Три вопроса, которые нужно задать после каждого исправления отклонения

Когда вы обнаружите, что Codex снова допустил ошибку, не ограничивайтесь фразой «в будущем будем внимательнее». Задайте три вопроса:

Является ли это случайным недоразумением в рамках данного задания или оно будет повторяться в будущем?

Требуется ли для этого текстовое правило, или это можно проверить автоматически?

Где следует разместить новое правило: в подсказках текущего раунда, в договоре о задании, в файле AGENTS.md, в тестовом доступе или в процессе утверждения?

Например, в одном приложении для Windows когда-то возникала ситуация: «компиляция прошла успешно, но при запуске происходил сбой». Если просто сказать в чате «в следующий раз не забудь запустить», этот опыт быстро забудется. Только включив проверку точного пути запуска, проверку журнала событий и проверку процессов очистки в постоянную процедуру проверки, можно превратить урок, извлечённый из ошибки, в актив проекта.

Самопроверка системы задач



[] Подсказки данного раунда содержат только необходимую на данный момент информацию;



[] для сложных задач уже подготовлены обновлённые соглашения о задачах и планы этапов;



[] долгосрочные правила уже включены в постоянную документацию на уровне репозитория;



[] В ключевых правилах для частей, поддающихся автоматической проверке, уже установлены ограничения доступа;



[] Отчеты о реализации, фактическом выполнении и внешнем состоянии подаются отдельно;



[] Существует по крайней мере один механизм проверки, способный опровергнуть вывод исполнителя о завершении;



[] Правила, обсужденные в ходе ретроспективы, размещены на соответствующем уровне, а не остаются только в чате.

Хорошие подсказки позволяют улучшить качество отве-

та за один раз. Система задач же позволяет постоянно меняющемуся проекту, даже после нескольких раундов совместной работы, по-прежнему понимать, с чего он начался, чего нельзя делать и какие доказательства считаются достижением цели.

Основания и границы данной главы



Примеры платформ взяты из реального проекта хостинговой платформы, разработанного автором, и соответствующих файлов доказательств (C05), а также из матрицы взаимодействия с примерами; в данной главе обобщены только части, относящиеся к системе задач, полные инженерные доказательства приведены в главе 25.



[1] Официальные материалы OpenAI Codex:

[«Codex Best Practices»](#)

и

[«Prompting](#)

».



AGENTS.md, планирование и проверка — это инструменты разных уровней; конкретные доступные возможности и интерфейс могут меняться в зависимости от версии Codex и используемого интерфейса, поэтому в рукописи приведены только относительно стабильные методы.

Часть 2: Превращение идей в выполнимые задачи

В этой части рассмотрено преобразование расплывчатых пожеланий в целевые результаты, реалистичные базовые условия, авторитетный контекст, границы, полномочия и полное задание.

Глава 5. Не начинайте со списка функций

Суть этой главы: функции — это варианты реализации, а не требования. Сначала определите, что должен сделать пользователь, из какого состояния он должен перейти в какое, а затем позвольте Codex выбрать способ реализации.

Список, выглядящий очень профессионально

Предположим, вы хотите создать небольшую логическую игру и предоставляете Codex список:

- область возможных ответов;
- шаги рассуждения;
- узлы доказательства;
- Пошаговые подсказки;
- Обратная связь при неудаче;
- Система звезд;
- полный анализ.

Этот список не содержит двусмысленностей. Codex может реализовать каждый пункт по отдельности, а тестирование — проверить каждый из них. Отчет о завершении будет выглядеть очень красиво.

Однако в списке не указано, что именно должен в конечном итоге сделать игрок. Должен ли он заполнить отчет о

расследовании или, наблюдая за тремя уликами, самостоятельно вывести ответ в уме?

Обе эти цели могут использовать «подсказки, обратную связь при неудаче и разбор», но приведут к совершенно разным результатам.

Именно таким образом отклонился первый уровень ранней версии игры «Игра ума». Старая версия имела 12 вариантов ответа, семишаговый анализ и доказательство процесса, а внутренний набор доказательств давал результат 12/12 PASS. Руководитель проекта, увидев экран, указал: дедуктивные выводы игрока должны происходить в его голове, а система должна принимать только окончательный ответ.

Как только результат пользователя становится ясным, многие «функциональные требования» сразу превращаются в препятствия, которые следует устранить.

Почему список функций так привлекателен

У функциональных требований есть три преимущества: их можно сосчитать, по ним можно распределить обязанности и их можно проверить.

«Добавить три страницы» — это более чётко, чем «упростить пользователю поиск контента»; «добавить процесс подтверждения» — легче реализовать, чем «сделать головоломку более глубокой»; «создать отдельную страницу поиска» больше похоже на инженерную задачу, чем «улучшить опыт просмотра скриншотов для опытных пользователей».

Поэтому, когда результат не определён, команде легко заменить прогресс количеством функций.

Проблема в том, что функции отвечают только на вопрос «что есть в системе», но не на вопрос «что благодаря этому может сделать пользователь».

Продукт может иметь больше страниц, более длинные процессы и более обширные настройки, но при этом отдалять пользователя от цели. Codex снижает механические затраты на добавление функций, что, в свою очередь, требует от людей более строгого обоснования необходимости каждой функции.

Формулируйте требования в виде изменений состояния

Выполнимый результат для пользователя должен включать как минимум пять элементов:

Пользователь: кто использует;

Ситуация: в какой момент, с какой проблемой;

Отправная точка: что в настоящее время невозможно выполнить на сайте или какую цену приходится за это платить;

Конечная цель: что можно сделать после завершения задачи;

Доказательства: как можно убедиться, что изменения действительно произошли.

Можно выразить одной фразой:

Когда [пользователь] сталкивается с [текущим препятствием] в [сценарии], он может [выполнить целевое действие], что подтверждается [доказательством результата], при этом соблюдая [ключевые границы].

Уровень с паролем можно описать так:

Когда игрок впервые запускает первый уровень, он должен без изучения дополнительных инструкций понять три подсказки, ввести полный пароль и получить общую обратную связь; проверка осуществляется с помощью скриншота одного экрана, полного взаимодействия и повторения иг-

роком при первом прохождении, при этом не раскрывая частично правильные ответы.

В этом предложении не указаны узлы Godot, названия элементов управления или структура кода, но его достаточно, чтобы отсеять кандидатов и подтвердить рабочую версию.

Целевое состояние должно также включать состояние неудачи

Описание только «что происходит в случае успеха» по-прежнему является неполным. Большая часть сложности реального продукта связана с: неполным вводом, сбоями в сети, конфликтами данных, недостаточными правами доступа, отменой действий пользователем и отсутствием ответа от внешних систем.

Целевое состояние уровня с вводом пароля включает не только «прохождение с правильным ответом», но и:

- отсутствие ошибочного определения результата при незаполненных трех цифрах;
- при полном неправильном ответе должен возвращаться только общий статус «неудача»;
- после перезапуска все введенные данные должны возвращаться в исходное состояние;
- отсутствие штрафов за пробы методом проб и ошибок в виде израсходования энергии, периода ожидания или обратного отсчёта.

То же самое касается поиска в ClipDock. «Возможность

поиска» — это не полное состояние; необходимо, как минимум, указать: скрытые элементы не должны выделяться рамкой; как обрезается исходное выделенное множество; что отображается при нулевом результате; сохраняется ли порядок рабочих областей после завершения поиска.

Состояние неудачи — это не обрывки, добавленные тестировщиком в последнюю минуту, а часть результата, получаемого пользователем.

Сначала спросите: «Можно ли повторно использовать?», а потом: «Что нужно добавить?»

ClipDock когда-то реализовал поиск в виде отдельной страницы результатов. Постепенно он обзавелся шестью видами сортировки, собственной синхронизацией выбора, контекстным меню и пакетными операциями в нижней части экрана — и выглядел всё более полноценным.

Но на самом деле пользователю нужно лишь одно: быстро найти нужный контент на текущей рабочей области скриншота и продолжить использовать привычные функции выделения, контекстного меню, просмотра деталей и пакетных команд.

После того как руководитель проекта отклонил второй вариант продукта, в Codex удалили отдельную страницу поиска и перенесли поиск в виде фильтрации прямо на главном холсте. В связи с этими изменениями было добавлено 992 строки и удалено 2337 строк.

Настоящий прогресс заключается не в том, что «уменьшилось количество кода на 1345 строк», а в том, что система вновь построена вокруг одного и того же основного цикла пользователя.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.